

LFMを使った高速インメモリ XMLデータベース Karearea



株式会社セック
マーケティング本部 土井 憲雄
doi@sec.co.jp

近年、インターネット技術の急速な発達と浸透により、PCだけでなくPDAや携帯電話などからWebやメールにアクセスするコミュニケーション手段が、ごく一般的に用いられるようになってきています。

1990年代後半に世界的かつ爆発的に広がったWorld Wide Webは、当初のHTMLを参照するのみの用途から、オンラインショッピングやバンキング、株取引などといった、複雑でインタラクティブな機能を持つようになってきました。また、その拡張性や開発生産性、保守性の高さのほか、オープンな技術であることなどから、企業内における基幹システムとしてだけでなく、企業間連携のシステムなどでも積極的に採用されるようになってきています。

一方、Webを使用したシステムが複雑で高度な機能を持つにつれ、もともと表示用として考案されたHTMLでは、複雑なデータの表現やシステム間での受け渡しなどが困難になってきています。

SGMLがももとなったXMLは、当初は文書のためのマークアップ言語として考えられていましたが、その柔軟性や拡張性から、最近ではシステム間連携の共通言語としての地位もほぼ確立しつつあります。今後は、コンピュータシステムやインターネットの世界だけでなく、あらゆる情報機器がXMLをベースにしてコミュニケーションするようになると考えられています。

1. RDBMSを使ったXMLの格納

XMLの活用範囲が広がるにつれ、大量のXMLデータをデータベースに格納したいというニーズも高くなってきています。しかし現在その目的で使用されているのは大半がRDBMS（リレーショナルデータベース管理システム）です。ところがRDBは2次元の表形式を基本として設計するため、木構造や半構造を持つXMLデータを効率よく格納することは、一般的に難しいことも

事実です。

ここで半構造とは、以下のような構造を指します。

- ・要素の繰り返し（同一タグを持つ要素が複数（不定期）現れる）
- ・要素の欠損（同一タグを持つ要素が、それを含む要素によっては存在しない）
- ・要素の選択肢（同一タグを持つ要素が、より下位のレベルで構造が異なる）

RDBにXMLデータを格納する場合の主な方式と、それぞれの長所・短所を「RDBを使ったXMLデータの格納処理方式の比較」（表1）に示します。

RDBMSベンダーやサードベンダー各社は、XMLを格納したいというニーズの高まりとともに、RDBMSにXML用のインタフェースを提供しています。しかし、もともと構造の異なるXMLデータをRDBの2次元の表にマッピングするため、多くは柔軟性を犠牲にして性能をとるか、あるいは性能を捨てて柔軟性をとるかの択一になっているのが現状です。

2. ネイティブXMLデータベースの登場

このような状況の中、最近ではXMLの格納や検索に特化したネイティブXMLデータベース製品が登場してきています。XMLに特化しているため、XMLを簡単に格納、検索して取り出す機能を持ち、XMLのデータ構造の変更などにも柔軟に対応することができます。

しかしネイティブXMLデータベース製品はまだ登場して間もなく、性能チューニングに関する技術やノウハウがあまり蓄積・確立されていません。RDBMS製品に比べると技術者もまだ少なく、SQLのような確立されたデータベースアクセス言語也没有ありません。（W3CにおいてXQueryなどが検討されていますが、まだ対応した製品もほとんど

表 1 RDBを使ったXMLデータの格納処理方式の比較

名称	格納方式	主な長所	主な短所
ファイル格納方式	XML データを文字列のままテーブルの 1 カラムに格納する	XML の構造変更があってもテーブル構造に影響を受けない Well-Formed な XML を格納することができる	基本的に全文検索になるため、範囲検索や複雑な条件での検索が難しい 個々の要素の集計処理などに不向き
テーブル写像方式	XML の各要素を、それぞれ対応するテーブルのカラムに格納する	個々の要素の検索や集計、更新などの処理性能が高い 既存の RDB 技術を利用できる	XML 構造とテーブルのマッピングが難しい XML の構造変更の影響が大きく、保守性や柔軟性に欠ける 半構造への対応が難しい
木構造写像方式	XML データをノードとリンクとに分解し、それぞれを管理するテーブル/カラムに展開する	XML の構造変更柔軟に対応できる 半構造に対応できる	アクセス方法が複雑になる 複雑な検索や大量データの集計処理などの高速化が困難

どなく、更新系についての検討もこれからという状況です)

また、XMLを採用するとデータに柔軟性や拡張性を持たせることができる一方、XMLを使ったシステムを構築した先進ユーザからは、データが冗長になり要求される性能を得るのに苦労する、という話を多く聞きます。

XMLの持つ柔軟性や拡張性を保持したまま、高速性能を実現することができないだろうか、と考えて開発したのが、LFMエンジンを使用したインメモリ高速XMLデータベース「Karearea (カレアレア)」です。

3. LFMを使ったXMLデータベース Karearea

LFMは基本的にRDBと同様、2次元の表形式のデータ構造を持っていますので、XMLデータを格納するためには前述のRDBへの格納と同様方式を選択することが考えられます。

弊社で開発したKareareaは、基本的には木構造写像方式と同様ですが、従来の方式とは異なる全く新しい格納方式を採用しています。それは一つの要素を一つのテーブルに対応させ、XMLの親子関係をテーブル間のリンク情報として表す方式です。

このデータ構造は、様々なXMLに柔軟に対応することができる半面、一般的なRDBMSを使って実装すると、データの検索や獲得の際に多段階に

わたる検索操作やJOIN操作が必要になり、性能的にはとても遅すぎて使い物にはなくなります。LFMエンジンでは高速な検索・JOIN操作が可能であるばかりでなく、従来のRDBMSでは備えていないような多様なデータ操作が用意されています。KareareaではこれらLFMエンジンの機能を駆使し、一般的なRDBMSとは大きく異なる操作を採用することによって、高速な検索や格納処理を実現しました。

さらにKareareaは、LFMエンジンが持つ高速なソート機能や多次元集計機能をそのままXMLデータに対して適用することを可能にしています。よってKareareaを使うことで、通常のXMLデータベースとしてだけでなく、XMLを使ったOLAPやデータマイニングなどの用途に適用することもできます。

一方、Kareareaにはいくつか制約もあります。

第一にKareareaはインメモリデータベースであるため、動作しているコンピュータをシャットダウンするとデータベース内のデータが消えてしまうこと、第二にすべてのXMLデータがメモリ空間上に載っている必要があること、第三にKareareaデータベースにXMLを格納する際にはスキーマ定義 (DTD) が必要であることです。

第一の対策としてKareareaは、高速Save/Load機能を備えています。高速Save/Load機能は、メモリ上に展開したイメージをそのままファイルに書き出し、またメモリ上に復元する機能で、数万

ドキュメントのXMLデータに対しても処理は数秒程度で完了します（処理時間はマシンの性能に依存します）。

第二の点については、LFMエンジンが持つデータ圧縮機能によりメモリの利用効率を高めており、実装メモリ容量が比較的小さくても快適に動作する工夫がなされています。しかしアドレッシングの限界値に制約されるため、32bit版では数千万件オーダーのノード数のデータ量を扱うことはできません（64bit版では、ほぼこれらの制約から解放されます）。

第三の点については、DTD不要のWell-FormedなXMLの中からどれだけ有意な検索を行うことが可能か疑問であることなどから、DTDを必須とする仕様を選択しました。その代わりDTDを利用したValidation機能を持っているため、データ登録時にXMLの妥当性を検証し、不正なデータを登録時に除外することが可能です（将来的にはXML SchemaやRelax NGにも対応予定です）。

4. Kareareaの性能

Kareareaの性能は、XMLデータの登録や削除に

においても他のRDBMSやXMLデータベース製品に比べて遜色はなく、むしろ優れています。Kareareaが最も優位性を発揮するのはやはり検索処理です。XMLの構造や問い合わせ条件などに大きく左右されることなく、常に高速な検索処理が行えます。

図1に示すDTDとXMLデータ（最大18,000ドキュメント/約32万ノード）を使用し、登録されたXMLデータの件数（ノード数）を変化させて全件検索および全件データ取得した場合の性能測定結果を、図2に示します（測定環境はCPU/Pentium III 800MHz、Memory/1GB、OS/Windows2000 Server）。

またXMLの階層の深浅を変えて性能測定を行った結果を表3に示します。KareareaがXMLの構造にあまり依存せず、性能が安定して高速であることがわかります。

5. まとめ

標準化の進展や適用範囲の拡大により今後急速に普及すると考えられているXMLですが、一般的にデータをXML化すると冗長になり、性能面では

図1 性能測定用のDTDとXML例

DTD	XML
<pre> <!ELEMENT stock-price (trading-date, issue-data)> <!ELEMENT trading-date EMPTY> <!ATTLIST trading-date year NMTOKEN #REQUIRED> <!ATTLIST trading-date month NMTOKEN #REQUIRED> <!ATTLIST trading-date day NMTOKEN #REQUIRED> <!ELEMENT issue-data (issue-category, stock-index, issue-code, issue-name, todays-start-price, todays-high-price, todays-low-price, todays-end-price, todays-output)> <!ELEMENT issue-category (#PCDATA)> <!ATTLIST issue-category code NMTOKEN #REQUIRED> <!ELEMENT stock-index EMPTY> <!ATTLIST stock-index nikkei (0 1) "0"> <!ATTLIST stock-index topix (0 1) "0"> <!ELEMENT issue-code (#PCDATA)> <!ELEMENT issue-name (#PCDATA)> <!ELEMENT todays-start-price (#PCDATA)> <!ELEMENT todays-high-price (#PCDATA)> <!ELEMENT todays-low-price (#PCDATA)> <!ELEMENT todays-end-price (#PCDATA)> <!ELEMENT todays-output (#PCDATA)> </pre>	<pre> <?xml version='1.0' encoding='Shift_JIS' ?> <stock-price> <trading-date year='2001' month='04' day='02' /> <issue-data> <issue-category code='11'>石油</issue-category> <stock-index nikkei='0' topix='0' /> <issue-code>1234</issue-code> <issue-name>カレアレア</issue-name> <todays-start-price>148</todays-start-price> <todays-high-price>151</todays-high-price> <todays-low-price>146</todays-low-price> <todays-end-price>148</todays-end-price> <todays-output>181000</todays-output> </issue-data> </stock-price> </pre>

図 2 全ノード数による検索時間変化

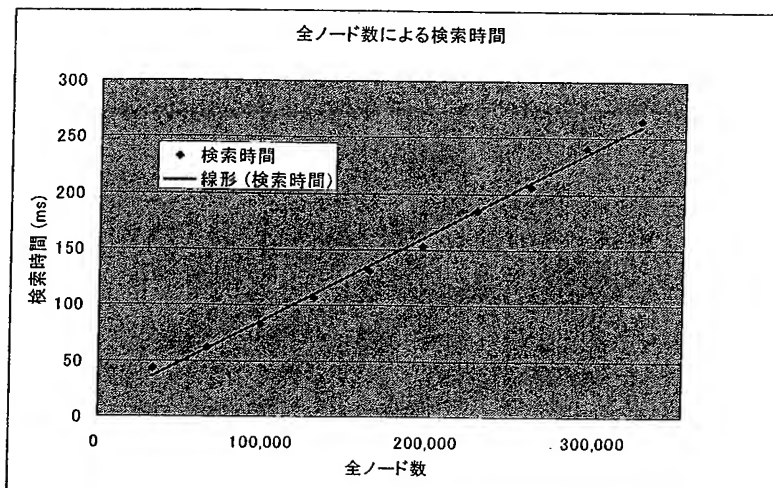


表 2 XMLの階層の深浅による性能測定結果

検索・取得条件	処理時間
浅い階層（2 階層）構造／全体で 10 万ノードの XML データから、検索条件に該当する 18 ドキュメントを抽出	9ms
深い階層（10 階層）構造／全体で 10 万ノードの XML データから、検索条件に該当する 18 ドキュメントを抽出	13ms

不利になります。

また、XMLをデータベースに格納する方式にはさまざまなものがあり、RDBやOODB（オブジェクト指向データベース）、ネイティブXMLデータベースという違いだけでなく、個々の製品によってもアプローチの仕方が異なります。そしてXMLの拡張性・柔軟性を保ったまま、高速な処理が行えるようなデータの格納・検索方式は非常に難しいことも事実で、いまなお各ベンダーがしのぎ

を削っています。

Kareareaは、高速な検索・ソート・JOIN処理が行えるLFMエンジンの利点を最大限に活かし、XMLの柔軟性と高速性能を両立させた数少ないインメモリXMLデータベースです。メモリ価格が安価になる一方、メモリ容量が大幅に増加しつつある現在、Kareareaのように高速に処理が行えるインメモリデータベースの需要は、今後ますます高くなって行くと考えられます。